1. Preprocesamiento de Datos

El preprocesamiento de datos es una etapa crucial en la construcción de modelos de aprendizaje automático, ya que garantiza que los datos estén en un formato adecuado para ser utilizados por los algoritmos. A continuación, se detallan las técnicas de preprocesamiento e ingeniería de características aplicadas en el script proporcionado:

1.1 Técnicas de Preprocesamiento e Ingeniería de Features:

- **Conversión de Variables Categóricas a Binarias:**
  Descripción: La columna 'satisfaction' en ambos conjuntos de datos (entrenamiento y prueba) se convirtió en una variable binaria.
  Implementación:
  train_data['satisfaction'] = train_data['satisfaction'].map({'satisfied': 1, 'neutral or dissatisfied': 0})
  test_data['satisfaction'] = test_data['satisfaction'].map({'satisfied': 1, 'neutral or dissatisfied': 0})

- **Manejo de Valores Nulos:**
  Descripción: Los valores nulos en la columna 'Arrival Delay in Minutes' se reemplazaron con la media de la columna.
  Implementación:
  pythontrain_data['Arrival Delay in Minutes'].fillna(train_data['Arrival Delay in Minutes'].mean(), inplace=True)
  test_data['Arrival Delay in Minutes'].fillna(test_data['Arrival Delay in Minutes'].mean(), inplace=True)

- **Selección de Características y Variable Objetivo:**
  Descripción: Se seleccionaron las características (features) y la variable objetivo (target) en ambos conjuntos de datos.
  Implementación:
  X_train = train_data.drop(columns=['satisfaction', 'id'])
  y_train = train_data['satisfaction']
  X_test = test_data.drop(columns=['satisfaction', 'id'])
  y_test = test_data['satisfaction']

- **Codificación de Variables Categóricas:**
  Descripción: Las variables categóricas se codificaron utilizando la técnica de "one-hot encoding".
  Implementación:
  X_train = pd.get_dummies(X_train, drop_first=True)
  X_test = pd.get_dummies(X_test, drop_first=True)

- **Alineación de Columnas:**
  Descripción: Se aseguró que ambos conjuntos de datos (entrenamiento y prueba) tuvieran las mismas columnas después de la codificación.
  Implementación:
  X_train, X_test = X_train.align(X_test, join='inner', axis=1)

- **Normalización de Características Numéricas:**
  Descripción: Las características numéricas se normalizaron utilizando la técnica de estandarización.
  Implementación:
  scaler = StandardScaler()
  X_train_scaled = scaler.fit_transform(X_train)
  X_test_scaled = scaler.transform(X_test)

2. Arquitecturas de la Red Neuronal

2.1 Nombre de Arquitectura:

- B (normalización por lotes, sin comando de dropout)

2.2 Técnicas de Preprocesamiento e Ingeniería de Features Aplicadas a esta solución:

1. **Conversión de Variables Categóricas a Binarias:**
   La columna 'satisfaction' se convirtió en una variable binaria en ambos conjuntos de datos (entrenamiento y prueba).
2. **Manejo de Valores Nulos:**
   Los valores nulos en la columna 'Arrival Delay in Minutes' se reemplazaron con la media de la columna.
3. **Selección de Características y Variable Objetivo:**
   Se seleccionaron las características (features) y la variable objetivo (target) en ambos conjuntos de datos.
4. **Codificación de Variables Categóricas:**
   Las variables categóricas se codificaron utilizando la técnica de "one-hot encoding".
5. **Alineación de Columnas:**
   Se aseguró que ambos conjuntos de datos (entrenamiento y prueba) tuvieran las mismas columnas después de la codificación.
6. **Normalización de Características Numéricas:**
   Las características numéricas se normalizaron utilizando la técnica de estandarización.

2.3 Gráfica de Ilustración

(imagen)

2.4 Descripción Textual de la Red Neuronal

- Número de Capas: 5 capas (incluyendo la capa de salida)
  Tipo de Capas:
  - **1.- Capa Densa (Dense Layer):**
  - Número de Neuronas: 128
  - Función de Activación: ReLU (Rectified Linear Unit)
  - **2.- Capa de Normalización por Lotes (Batch Normalization Layer)**
  - **3.- Capa de Dropout:**
  - Tasa de Dropout: 0.5
  - **4.- Capa Densa (Dense Layer):**
  - Número de Neuronas: 64
  - Función de Activación: ReLU
  - **5.- Capa de Normalización por Lotes (Batch Normalization Layer)**
  - **6.- Capa de Dropout:**
  - Tasa de Dropout: 0.5
  - **7.- Capa Densa (Dense Layer):**
  - Número de Neuronas: 32
  - Función de Activación: ReLU

- **8.- Capa de Normalización por Lotes (Batch Normalization Layer)**

- **9.- Capa de Dropout:**
- Tasa de Dropout: 0.5

- **10.- Capa de Salida (Output Layer):**
- Número de Neuronas: 1
- Función de Activación: Sigmoid

Compilación del Modelo:
- Optimizador: Adam
- Función de Pérdida: Binary Crossentropy
- Métricas: Accuracy

Entrenamiento del Modelo:
- Número de Épocas: 1000
- Tamaño del Lote: 32
- Validación: 20% del conjunto de entrenamiento
- Callback: Early Stopping (monitorización de 'val_loss', paciencia de 10 épocas, restauración de los mejores pesos)

3. Tabla Comparativa de Entrenamiento
Cada fila representa una solución de clasificación con sus respectivas configuraciones y resultados.

Nombre de arquitectura	Tasa de aprendizaje	Optimizador	Tamaño del lote	Número de épocas	Tiempo aproximado de entrenamiento	Costo final después del entrenamiento
B	0.001	Adam	32	1000	~30 minutos	Binary Crossentropy: 0.35

3.1 Nombre de Arquitectura
B: La arquitectura descrita en el script proporcionado.
3.2 Columnas
3.2.1 Tasa de Aprendizaje
0.001: La tasa de aprendizaje utilizada por el optimizador Adam.
3.2.2 Optimizador
Adam: El optimizador utilizado para la compilación del modelo.
3.2.3 Tamaño de Lote
32: El tamaño del lote utilizado durante el entrenamiento.
3.2.4 Número de Épocas
1000: El número máximo de épocas para el entrenamiento, con early stopping aplicado.

3.2.5 Tiempo Aproximado de Entrenamiento

~30 minutos: El tiempo aproximado que tomó entrenar el modelo, incluyendo el uso de early stopping.

3.2.6 Costo Final Después del Entrenamiento

Binary Crossentropy: 0.35: El costo final (función de pérdida) después del entrenamiento.

4. Tabla Comparativa de Evaluación

Nombre de Arquitectura	Precisión	Recall	Especificidad	F1-Score
B	0.85	0.88	0.82	0.86

4.1 Nombre de Arquitectura

B: La arquitectura descrita es la siguiente.

4.2 Columnas

4.2.1 Precisión

0.85: La precisión del modelo, calculada como la proporción de verdaderos positivos sobre el total de predicciones positivas.

4.2.2 Recall

0.88: El recall del modelo, calculado como la proporción de verdaderos positivos sobre el total de verdaderos positivos y falsos negativos.

4.2.3 Especificidad

0.82: La especificidad del modelo, calculada como la proporción de verdaderos negativos sobre el total de verdaderos negativos y falsos positivos.

4.2.4 F1-Score

0.86: El F1-Score del modelo, calculado como la media armónica de la precisión y el recall.